Particle Swarm Optimized Autonomous Learning Fuzzy System

Gu, Xiaowei; Shen, Qiang; Angelov, Plamen Parvanov

Published in:
IEEE Transactions on Cybernetics

DOI:
10.1109/TCYB.2020.2967462

Publication date:
2021

Citation for published version (APA):
Gu, X., Shen, Q., & Angelov, P. P. (2021). Particle Swarm Optimized Autonomous Learning Fuzzy System. *IEEE Transactions on Cybernetics*, *51*(11), 5352-5363. https://doi.org/10.1109/TCYB.2020.2967462

Particle Swarm Optimized Autonomous Learning Fuzzy System

Xiaowei Gu, Qiang Shen and Plamen P. Angelov, *Fellow, IEEE*

***Abstract*—The antecedent and consequent parts of a first-order evolving intelligent system (EIS) determine the validity of the learning results and overall system performance. Nonetheless, the state-of-the-art techniques mostly stress on the novelty from the system identification point of view but pay less attention to the optimality of the learned parameters. Using the recently introduced autonomous learning multiple model (ALMMo) system as the implementation basis, this paper introduces a particle swarm-based approach for EIS optimization. The proposed approach is able to simultaneously optimize the antecedent and consequent parameters of ALMMo and effectively enhance the system performance by iteratively searching for optimal solutions in the problem spaces. In addition, the proposed optimization approach does not adversely influence the "one pass" learning ability of ALMMo. Once the optimization process is complete, ALMMo can continue to learn from new data to incorporate unseen data patterns recursively without a full retraining. Experimental studies with a number of real-world benchmark problems validate the proposed concept and general principles. It is also verified that the proposed optimization approach can be applied to other types of EISs with similar operating mechanisms.**

***Index Terms*—Autonomous learning, evolving intelligent system, optimality, particle swarm optimization**

I. Introduction

EVOLVING intelligent systems (EISs) [1], [2] are capable of effectively approximately modeling non-stationary problems in real time. In particular, they have been widely used in real world applications for streaming data processing [3], [4]. EISs self-organize and gradually self-develop their system structure and parameters through "one pass" learning process from data streams. Comparing with neural networks or deep learning, EISs are able to efficiently transform streaming data into knowledge in a human-interpretable form, and they also demonstrate stronger ability of handling imprecisely described data patterns that are distinct from historical data.

One major issue of current EISs comes from the fact that both the antecedent and consequent parts of the system learned from streaming data are often not optimal. This is caused by the "one pass" learning procedure without involving any optimization mechanism, thereby being unable to iteratively search for the optimal solution. In order to address this issue and enhance the performance of EISs, a global searching technique is necessary.

X. Gu and P. Angelov are with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK, e-mail:{x.gu3,p.angelov}@lancaster.ac.uk

Q. Shen is with the Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, U.K. email: qqs@aber.ac.uk

Corresponding author: Xiaowei Gu

Manuscript received XXXX XX, 2019; revised XXXX XX, 2019.

Evolutionary computation (EC) techniques, inspired by biological evolution, are generally applied to global optimization. Particle swarm optimization (PSO) [5], [6] is such a technique that simulates the social behavior of animal flocking, and that has demonstrated strong ability in searching for optimal solutions in a range of problem spaces. Thanks to its conceptual simplicity and high computational efficiency, PSO has gained increasing popularity since its inception and has become one of the main tools for solving real-world optimization problems.

In this paper, a PSO-based approach is proposed for EISs to autonomously attain optimal solutions given imprecisely described problems. Using the recently introduced autonomous learning multiple model (ALMMo) fuzzy system [7] as the underlying implementation, the algorithmic procedure for simultaneously optimizing both the antecedent and consequent parameters of EISs is presented. This optimization process works on data samples collected during the online learning process. It helps significantly boost the performance of EISs in terms of the prediction accuracy owing to PSO's strong global searching ability. Numerical examples on benchmark problems as well as real-world ones are conducted for verifying the validity and effectiveness of the proposed general concept and principles.

The unique contribution of this paper is reflected by the following aspects.

1) A canonical PSO-based approach for optimizing both antecedent and consequent parts of EISs simultaneously based on historical observations only; and
2) A learning mechanism for the optimized EISs to learn from new observations in a "one pass" manner after optimization to efficiently incorporate new data patterns.

Whilst ALMMo is herein utilized to implement the proposed approach, the underlying techniques are generic and can be applied to other types of EIS with similar operating mechanisms. In addition, more advanced PSO algorithms can be employed to further strengthen the performance.

The remainder of this paper is organized as follows. Section II outlines the background of this study. Section III briefly recalls the learning process of ALMMo. The proposed PSO-based optimization algorithm is described in detail in Section IV. Section V presents numerical examples serving as the proof of the concept. This paper is concluded by Section VI and directions for future work are also given in this section.

II. Background

A. EISs

Nowadays, EISs have been widely used for streaming data processing in various application scenarios [4], [8], [9]. EISs

can be implemented in the forms of neuro-fuzzy [10]–[13] or fuzzy rule-based models [14]–[16]. Being an intensively studied area, a number of successful (neuro-) fuzzy models have been introduced, which include, but are not limited to, DENFIS [11], eTS [14], CNFS [12], PENsemble [13], FLEXFIS [16], pClass [15], SAFIS [17], McIT2FIS [18], eT2Class [19], PANFIS [20] and GENEFIS [21]. Due to the limited space of this paper, it is practically impossible to cover all the existing zero-order, first-order and higher-order (type-2) EISs and their applications. Interested readers are referred to the recent surveys [1], [2] for more details regarding EISs. This paper is focused on first-order EISs given their popularity and status of being the best developed in the literature.

Current first-order EISs usually employ different operating mechanisms for antecedent (IF) part identification. Widely-used rule evolving schemes include density criterion [14], [22], distance criterion [16] error criterion [17] and statistical contribution criterion [21]. These evolving schemes have different pros and cons, and there exist EISs incorporating several criteria together for better evolving ability and performance [7]. For the consequent parameter learning, the majority of EISs utilize recursive least squares [23] or its extensions, e.g. fuzzily weighted recursive least squares (FWRLS) [14] and maximum correntropy criterion [12]. There are also EISs that use extended Kalman filter for antecedent and consequent parameter updating [17], but such an optimization algorithm increases the computational burden while performing online learning.

Most of EISs learn from data in a "one pass" manner. They assemble prediction models from data samples coming in a stream form and continuously self-improve the performance by adapting both system structure and parameters based on feedback on their predictions. Existing EISs typically stress on the ability of reacting promptly to the shifts and/or drifts of underlying patterns of the streaming data [24]. This ability is, indeed, of great importance to the success of EISs due to the non-linear, non-stationary nature of data streams in real-world applications [1]. However, it also frequently leads to the poor global prediction accuracy and the so-called "unlearning effect" [25]. Moreover, both antecedent and consequent parameters of EISs often lack optimality due to the fact that the system structure and parameters are usually self-developed in an exploratory manner. The absence of parameter optimization process during the learning process of EISs may result in a significant loss of information carried by training data. As a result, the performance of EISs in terms of prediction accuracy can be further improved to a greater extent once the optimality issue is addressed.

Recently, increasingly more attention has been paid to the optimality and stability analysis of EISs [26]. The stability of the consequent (THEN) part of EISs has been shown [27]. In particular, a local error optimization approach for fuzzy system identification has been proposed in [25]. However, this is, again, a"one pass" approach and it lacks a theoretical proof of optimality. In [28], the optimality of EISs is systematically studied, and two algorithms are introduced for optimizing the antecedent and consequent parts, respectively. Nonetheless, the prediction accuracy is only improved marginally on certain problems after system optimization. This is mainly because the antecedent and consequent parts of the system are optimized separately, based on the use of different criteria [28]. The overall performance of an EIS is decided by both antecedent and consequent parts. To achieve the best global prediction accuracy, the two parts should be optimized as an entirety.

B. PSO

PSO is a population-based, heuristic and evolution algorithm firstly introduced by Kennedy and Eberhart [5], [6]. In PSO, a population of particles cooperate and interact to search for solutions in a certain problem space, by learning from individual best experience and the global best-so-far solution of the entire swarm. Because of its implementational simplicity and high computational efficiency, PSO has become a widely used optimization tool for various research and engineering problems. In the past two decades, a number of variants to the original have been introduced attempting to further improve its performance. Typical variants can be summarized in the following four types: (1) neighborhood topology [29], [30]; (2) parameter control [31], [32]; (3) hybrid methods [33], [34] and (4) novel learning schemes [35], [36]. Since genetic algorithms (GAs) have good exploration ability, genetic learning PSO (GLPSO) has been proposed [37] to strength the performance of PSO by generating high-quality exemplars to guide the evolution of the particles [38].

Currently, PSO algorithms have been successfully applied to many problems, e.g., clustering [39], control [40], feature selection and image processing [41], [42]. There also exist a number of approaches that involve PSO for training fuzzy systems [43]–[45] and artificial neural networks [46]. However, such approaches are limited to offline scenarios and require a full retraining if new data patterns occur. Therefore, they are restricted in applications that concern online"data rich" environments.

III. ALMMo* System

In this section, the general architecture and autonomous learning process of the ALMMo system are briefly recalled to make this paper self-contained. Note, however, that the ALMMo used in this paper differs from the original version [7] in the following two aspects:

1) The firing strength of each fuzzy rule is calculated using Gaussian kernel function; and
2) The processed historical data are stored in a data pool instead of being discarded.

Gaussian kernel function is used by most of first-order EISs [12], [25]. Comparing with Cauchy kernel function (which is originally adopted in [7]), Gaussian is more compact and sensitive to outliers and thus, it helps improve the ability of ALMMo system to handle unfamiliar data patterns, enhancing its online learning capability. The historical data saved in the data pool will be used for system parameter optimization. To distinguish the current version from the original, the ALMMo system used herein is denoted as ALMMo*.

A. General Architecture

The structure of ALMMo* is depicted in Fig. 1. The system is composed of N linear models identified through its inherent learning process. Each linear model is described by a prototype-based fuzzy rule as described in Eqn. (1):

$$\boldsymbol{R}_i: \quad IF\ (\boldsymbol{x} \sim \boldsymbol{p}_i)\ THEN\ (y_i = \bar{\boldsymbol{x}}^T \boldsymbol{a}_i); \tag{1}$$

where $i = 1, 2, ..., N$; N is the number of linear models (fuzzy rules); $\boldsymbol{x} = [x_1, x_2, ..., x_M]^T \in \Re^M$ is the input vector of the fuzzy rule; $\bar{\boldsymbol{x}} = [1, \boldsymbol{x}^T]^T$; $\Re^M$ is an M dimensional real data space; y_i is the output of $\boldsymbol{R}_i$; $\boldsymbol{p}_i = [p_{i,1}, p_{i,2}, ..., p_{i,M}]^T$ and $\boldsymbol{a}_i = [a_{i,0}, a_{i,1}, ..., a_{i,M}]^T$ are the prototype and consequent parameters, respectively.

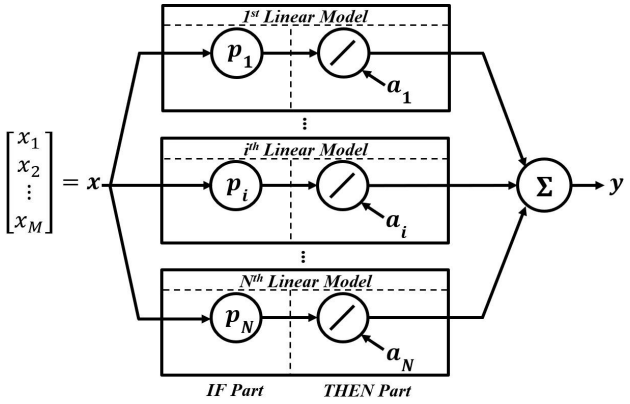

Fig. 1: System structure of ALMMo*.

The overall ALMMo* system is mathematically modeled by

$$y = f(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i y_i = \sum_{i=1}^{N} \lambda_i \bar{\boldsymbol{x}}^T \boldsymbol{a}_i; \tag{2}$$

where λ_i is the firing strength of the i^{th} fuzzy rule, $\boldsymbol{R}_i$ such that

$$\lambda_i = \frac{D_i(\boldsymbol{x})}{\sum_{j=1}^{N} D_j(\boldsymbol{x})}; \tag{3}$$

here $D_i(\boldsymbol{x})$ represents a Gaussian kernel-based local density of $\boldsymbol{x}$ calculated within the data cloud, $\mathbf{C}_i$ that is formed around $\boldsymbol{p}_i$ by nearby data samples resembling Voronoi tessellation [47]:

$$D_i(\boldsymbol{x}) = e^{-\frac{||\boldsymbol{x}-\hat{\boldsymbol{p}}_i||^2}{2(\hat{X}_i - ||\hat{\boldsymbol{p}}_i||^2)}}. \tag{4}$$

In Eqn. (4), $||\cdot||$ denotes the Euclidean norm; $\hat{\boldsymbol{p}}_i$ and $\hat{X}_i$ are calculated by:

$$\hat{\boldsymbol{p}}_i = \frac{S_i}{S_i+1}\boldsymbol{p}_i + \frac{1}{S_i+1}\boldsymbol{x}; \tag{5a}$$

$$\hat{X}_i = \frac{S_i}{S_i+1}X_i + \frac{1}{S_i+1}||\boldsymbol{x}||^2; \tag{5b}$$

where $\boldsymbol{p}_i$ is the prototype of $\boldsymbol{R}_i$ and $\boldsymbol{p}_i = \frac{1}{S_i}\sum_{\boldsymbol{x}\in\mathbf{C}_i}\boldsymbol{x}$; X_i is the mean of squared Euclidean norm of data samples affiliated with $\boldsymbol{p}_i$ and $X_i = \frac{1}{S_i}\sum_{\boldsymbol{x}\in\mathbf{C}_i}||\boldsymbol{x}||^2$; and S_i is the cardinality (number of members) of $\mathbf{C}_i$.

Firing strength calculated as per Eqn. (4) allows ALMMo* to react to potential changes in data patterns promptly because it enlarges the role of any new observation, $\boldsymbol{x}$ by attracting $\boldsymbol{p}_i$ towards it. This enhancess the adaptive ability of the system.

B. Online Learning Process

The online autonomous learning process of ALMMo* is summarized below.

Stage 1. System Initialization

The ALMMo* system is initialized by the first observed data sample, $\boldsymbol{x}_k$ $(k = 1)$. The global meta-parameters of the system, global mean, $\boldsymbol{\mu}$ and average squared Euclidean norm, χ are set as:

$$\boldsymbol{\mu} \leftarrow \boldsymbol{x}_k;\ \chi \leftarrow ||\boldsymbol{x}_k||^2. \tag{6}$$

The first data cloud, $\mathbf{C}_N$ $(N = 1)$ is initialized with $\boldsymbol{x}_k$ as its prototype:

$$\begin{aligned} \mathbf{C}_N \leftarrow \{\boldsymbol{x}_k\};\ \boldsymbol{p}_N \leftarrow \boldsymbol{x}_k;\ X_N \leftarrow ||\boldsymbol{x}_k||^2; \\ S_N \leftarrow 1;\ \Lambda_N \leftarrow 0;\ \eta_N \leftarrow 1;\ I_N \leftarrow k; \end{aligned} \tag{7}$$

where Λ_N denotes the accumulated firing strength; η_N is the utility; and I_N is the time instance at which $\boldsymbol{p}_N$ is identified.

By initializing the consequent parameters, $\boldsymbol{a}_N$ and converiance matrix, $\boldsymbol{\Theta}_N$ using Eqn. (8) and (9),

$$\boldsymbol{a}_N \leftarrow \mathbf{0}_{(M+1)\times 1};\ \boldsymbol{\Theta}_N \leftarrow \Omega_o \mathbf{I}_{(M+1)\times(M+1)}, \tag{8}$$

the first fuzzy rule within the rule base is established such that:

$$\boldsymbol{R}_N: \quad IF\ (\boldsymbol{x} \sim \boldsymbol{p}_N)\ THEN\ (y_N = \bar{\boldsymbol{x}}^T \boldsymbol{a}_N), \tag{9}$$

where $N = 1$. Note that Ω_o is an externally controlled parameter for covariance matrix initialization, representing the standard for recursive least squares algorithms [48]. The first data sample and the desired output, $\{\boldsymbol{x}_k, y_k\}$ are then stored in the data pool for future use.

Stage 2. System Output Generation

When a new data sample, $\boldsymbol{x}_k$ $(k \leftarrow k+1)$ is observed, the local density, $D_i(\boldsymbol{x}_k)$ $(i = 1, 2, ..., N)$ of $\boldsymbol{x}_k$ at each data clould, $\mathbf{C}_i$ is firstly calculated using (4). Then, the firing strength, $\lambda_i (i = 1, 2, ..., N)$ of each fuzzy rule is calculated with respect to Eqn. (3) and the system output $\hat{y}_k = f(\boldsymbol{x}_k)$ is generated using Eqn. (2).

Stage 3. Global Meta-Parameter Updating

The global mean, $\boldsymbol{\mu}$ and average squared Euclidean norm, χ are updated using Eqn. (10a) and (10b) below, respectively:

$$\boldsymbol{\mu} \leftarrow \frac{k-1}{k}\boldsymbol{\mu} + \frac{1}{k}\boldsymbol{x}_k; \tag{10a}$$

$$\chi \leftarrow \frac{k-1}{k}\chi + \frac{1}{k}||\boldsymbol{x}_k||^2. \tag{10b}$$

Stage 4. System Structure Updating

In this stage, firstly, the global densities at $\boldsymbol{x}_k$ and $\{\boldsymbol{p}\}_N$ are calculated by Eqn. (11) (in a similar form to Eqn. (4)):

$$D(\boldsymbol{z}) = e^{-\frac{||\boldsymbol{z}-\boldsymbol{\mu}||^2}{2(\chi - ||\boldsymbol{\mu}||^2)}}; \tag{11}$$

where $\boldsymbol{z} \in \{\boldsymbol{x}_k, \boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_N\}$.

Condition 1 is checked to see whether $\boldsymbol{x}_k$ has the potential to be a new prototype:

$$\begin{aligned} \textbf{Condition 1}: \ & If\ (D(\boldsymbol{x}_k) < \min_{i=1,2,...,N}(D(\boldsymbol{p}_i))) \\ & Or\ (D(\boldsymbol{x}_k) > \max_{i=1,2,...,N}(D(\boldsymbol{p}_i))) \\ & Then\ (\boldsymbol{x}_k\ is\ a\ new\ prototype) \end{aligned} \tag{12}$$

If $D(\boldsymbol{x}_k)$ is greater than the global density calculated with respect to the existing prototypes, $\{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_N\}$, it means that $\boldsymbol{x}_k$ is more representative and has stronger summarization power than other prototypes. On the other hand, if $D(\boldsymbol{x}_k)$ is smaller than the minmim global density calculated with the prototypes, $\boldsymbol{x}_k$ stands for a new data pattern that is very different from all the previously identified ones. Therefore, if **Condition 1** is met in either case, $\boldsymbol{x}_k$ becomes a new prototype and initializes a new data cloud.

Once **Condition 1** is satisfied, **Condition 2** is checked to remove the nearby data cloud whose area of influence heavily overlaps with the new data cloud initialized by $\boldsymbol{x}_k$:

$$\begin{aligned} &\textbf{Condition 2}: \ \textit{If } (D_i(\boldsymbol{x}_k) \geq D_o) \\ &\textit{Then } (\boldsymbol{x}_k \textit{ is very close to } \boldsymbol{p}_i) \end{aligned} \tag{13}$$

where $D_o = e^{-\frac{1}{8}}$. The rationale behind **Condition 2** is that if $D_i(\boldsymbol{x}_k)$ is larger than $e^{-\frac{1}{8}}$, the Euclidean distance between $\boldsymbol{x}_k$ and $\boldsymbol{p}_i$ is smaller than half of the average distance between any two data samples within $\mathbf{C}_i$. In such case, $\boldsymbol{x}_k$ and $\boldsymbol{p}_i$ are very close and therefore, intuitively $\boldsymbol{p}_i$ should be replaced by $\boldsymbol{x}_k$ to avoid redundancy [7].

If both **Conditions 1** and **2** are met, the nearest data cloud denoted by $\mathbf{C}_{n*}$ regarding $\boldsymbol{x}_k$ is identified by the following equation:

$$n* = \underset{i=1,2,...,N}{\arg\min} (||\boldsymbol{x}_k - \boldsymbol{p}_i||), \tag{14}$$

and $\boldsymbol{x}_k$ is assigned to $\mathbf{C}_{n*}$ with its meta-parameters updated such that:

$$\begin{aligned} \mathbf{C}_{n*} \leftarrow \mathbf{C}_{n*} \cup \{\boldsymbol{x}_k\};\ \boldsymbol{p}_{n*} \leftarrow \frac{\boldsymbol{p}_{n*} + \boldsymbol{x}_k}{2}; \\ X_{n*} \leftarrow \frac{X_{n*} + ||\boldsymbol{x}_k||^2}{2};\ \boldsymbol{S}_{n*} \leftarrow \left\lceil \frac{\boldsymbol{S}_{n*} + 1}{2} \right\rceil; \end{aligned} \tag{15}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation.

If only **Condition 1** is met, a new data cloud with $\boldsymbol{x}_k$ as its prototype is added to the system ($N \leftarrow N + 1$) following Eqn. (7) and a new fuzzy rule $\boldsymbol{R}_N$ is initialized in the same form as Eqn. (9) with the consequent parameters initialized accordingly by Eqn. (16) below:

$$\boldsymbol{a}_N \leftarrow \frac{1}{N-1} \sum_{i=1}^{N-1} \boldsymbol{a}_i;\ \boldsymbol{\Theta}_N \leftarrow \Omega_o \cdot \mathbf{I}_{(M+1)\times(M+1)}. \tag{16}$$

Otherwise, if **Condition 1** is unsatisfied, $\boldsymbol{x}_k$ is assigned to the nearest data cloud, $\mathbf{C}_{n*}$ with its meta-parameters updated with respect to Eqn. (17):

$$\begin{aligned} S_{n*} \leftarrow S_{n*} + 1;\ \boldsymbol{p}_{n*} \leftarrow \frac{S_{n*} - 1}{S_{n*}} \boldsymbol{p}_{n*} + \frac{1}{S_{n*}} \boldsymbol{x}_k; \\ X_{n*} \leftarrow \frac{S_{n*} - 1}{S_{n*}} X_{n*} + \frac{1}{S_{n*}} ||\boldsymbol{x}_k||^2; \end{aligned} \tag{17}$$

For the data clouds that do not receive new members at the current processing cycle, their meta-parameters stay the same.

Stage 5. Rule Base Quality Monitoring

In this stage, the firing strength of each fuzzy rule, λ_i ($i = 1, 2, ..., N$) is firstly calculated using Eqn. (3). Then, the accumulated firing strength of each fuzzy rule is updated using Eqn. (18) below:

$$\Lambda_i \leftarrow \Lambda_i + \lambda_i; \tag{18}$$

and the corresponding utility is updated by

$$\eta_i \leftarrow \frac{\Lambda_i}{k - I_i}. \tag{19}$$

Then, **Condition 3** is checked to remove any fuzzy rules and the corresponding data clouds that contribute little to the overall system output:

$$\begin{aligned} &\textbf{Condition 3}: \ \textit{If } (\eta_i < \eta_o) \\ &\textit{Then } (\boldsymbol{R}_i \textit{ and } \mathbf{C}_i \textit{ are removed}) \end{aligned} \tag{20}$$

where η_o is another externally defined parameter for fuzzy rule quality monitoring.

If $\boldsymbol{R}_i$ and $\mathbf{C}_i$ satisfy **Condition 3** and are removed from ALMMo*, $N \leftarrow N - 1$, and the firing strengths of the remaining fuzzy rules are re-calculated using Eqn. (3).

Stage 6. Consequent Parameter Updating

The consequent parameters of the fuzzy rules are updated using the popular FWRLS method as given in [14] ($i = 1, 2, ..., N$):

$$\boldsymbol{\Theta}_i \leftarrow \boldsymbol{\Theta}_i - \frac{\lambda_i \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k \bar{\boldsymbol{x}}_k^T \boldsymbol{\Theta}_i}{1 + \lambda_i \bar{\boldsymbol{x}}_k^T \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k}; \tag{21a}$$

$$\boldsymbol{a}_i \leftarrow \boldsymbol{a}_i + \lambda_i \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k (y_k - \bar{\boldsymbol{x}}_k^T \boldsymbol{a}_i). \tag{21b}$$

Stage 7. Rule Base Updating

In the final stage of the current processing cycle, all fuzzy rules within the rule base are updated correspondingly with regard to the latest prototypes and consequent parameters, and the current data sample and the corresponding desired output, $\{\boldsymbol{x}_k, y_k\}$ are stored in the data pool for future use. Then, ALMMo* goes back to Stage 1 ready for the next new data sample. The flowchart of the learning process of ALMMo* is presented in Fig. 2 to support illustration.

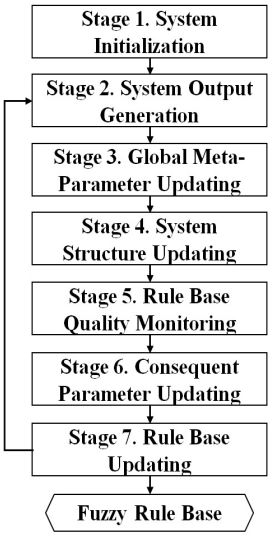

Fig. 2: Flowchart of ALMMo* learning process.

The two externally controlled parameters, Ω_o in Eqn. (8) and η_o in Eqn. (20) can subtly influence the performance of ALMMo system. Ω_o is for initializing the covariance matrix and η_o concerns the quality of fuzzy rules. Empirically, Ω_o influences the convergence of the consequent part of the fuzzy

system and may deteriorate the system performance if it is set too large or too small. Also, η_o may subtly influence the system structure. A larger value of η_o makes the system remove stale rules faster and vice versa. However, the system performance will deteriorate significantly if η_o is set too large since the system forgets the learned knowledge from historical data rather rapidly. The influence of Ω_o and η_o has been analyzed and verified through numerical examples in [7], [28]. The recommended values of Ω_o and η_o are 10 and 0.1, respectively.

IV. Optimizing ALMMo* by PSO

ALMMo* as well as the majority of other first-order type-1 EISs are computationally efficient thanks to the "one pass" type learning process. However, such learning process often leads to the problem that the solution learned from data lacks optimality. This can significantly and adversely influence the overall accuracy of the system. To address this issue, a PSO-based EIS optimization algorithm is proposed here for simultaneously optimizing both the premise and consequent parts of ALMMo*. Without loss of generality, the canonical single-objective PSO algorithm is employed for demonstrating the proposed concept. Nonetheless, if preferred, more advanced PSO algorithms may be used as alternative, (e.g., GLPSO [37], and multi-objective PSO algorithms [49]), but this is beyond the scope of this paper.

The algorithmic procedure of the proposed approach for EIS optimization consists of the following main steps:

Stage 1. Swarm Initialization

Firstly, the swarm $\mathbf{S}^t$, which is composed of L particles $\mathbf{S}^t = \{\mathbf{P}_1^t, \mathbf{P}_2^t, ..., \mathbf{P}_L^t\}$, is initialized, where t represents the current iteration ($t \leftarrow 0$). Each particle contains a full set of antecedent and consequent parameters of the fuzzy rules identified during the online learning process representing a full solution to the given problem despite that it may not be optimal. As PSO searches the data space in a semi-random manner, the leading particle, $\mathbf{P}_1^t$ can be formulated by Eqn. (22) to guarantee that the performance of ALMMo* does not deteriorate after the entire process is completed:

$$\mathbf{P}_1^t = [\mathbf{p}_1^t, \mathbf{a}_1^t]_{N\times 2(M+1)}, \tag{22}$$

where $\mathbf{p}_1^t = [\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_N]^T$ is the $N \times M$ dimensional antecedent parameter matrix (prototypes); and $\mathbf{a}_1^t = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_N]^T$ is the $N \times (M+1)$ dimensional consequent parameter matrix.

The maximum range that other particles can travel from the position of the leading particle, $\mathbf{P}_1$ within the searching space $\mathfrak{R}^{N\times 2(M+1)}$, during the initialization stage, is defined by:

$$\mathbf{Q} = [\boldsymbol{\beta}^T, \boldsymbol{\beta}^T, ..., \boldsymbol{\beta}^T]_{N\times 2(M+1)}^T \tag{23}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_{2(M+1)}]_{1\times 2(M+1)}$ and there is:

$$\beta_j = \max_{i=1,2,...,N}(|\mathbf{P}_1^t(i,j)|), \tag{24}$$

with $\mathbf{P}_1^t(i,j)$ denoting the element at the i^{th} row and j^{th}column of $\mathbf{P}_1^t$. The remaining particles are defined as ($i = 2, 3, ..., L$):

$$\mathbf{P}_i^t = [\mathbf{p}_i^t, \mathbf{a}_i^t]_{N\times 2(M+1)}^T = \mathbf{P}_1^t + \mathbf{r}_i \circ \mathbf{Q}, \tag{25}$$

where "$\circ$" represents Hadamard product; $\mathbf{r}_i$ is a $N \times 2(M+1)$ dimensional random matrix whose elements follow the uniform distribution within the value range of $[-1, 1]$; Eqn. (23) defines a local searching range surrounding the leading particle, $\mathbf{P}_1^t$ and Eqn. (25) scatters other particles randomly in the defined local area. The searching efficiency is thus, significantly improved thanks to the confined searching space.

To evaluate the fitness of $\mathbf{P}_i^t$ ($i = 1, 2, ..., L$), the data space needs to be re-partitioned by the use of $\mathbf{p}_i^t$ as prototypes to attract nearby historical data samples forming Voronoi tessellations using Eqn. (14). This results in N new data clouds, $\mathbf{C}_{i,j}^t$ ($j = 1, 2, ..., N$). Subsequently, the meta-parameters of each data cloud, which include cardinality, $S_{i,j}^t$, mean/prototype, $\boldsymbol{p}_{i,j}^t$, and average square Euclidean norm $X_{i,j}^t$, are extracted. With these updated meta-parameters ($\{\boldsymbol{p}\}_i^t$, $\{X\}_i^t$, $\{S\}_i^t$ and $\{\boldsymbol{a}\}_i^t$), the fitness of a particle is calculated by the following objective function:

$$F(\mathbf{P}_i^t) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}|y_k - f_i^t(\boldsymbol{x}_k)|^2}, \tag{26}$$

where $f_i^t(\boldsymbol{x})$ is the mathematical model of ALMMo* same as Eqn. (2) but with the antecedent and consequent parameters derived directly from $\mathbf{P}_i^t$. This objective function, $F(\mathbf{P}_i^t)$, essentially, calculates the root mean square error (RMSE) between the outputs of the ALMMo* system (obtained so far) using historical training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\}$ as the inputs and the corresponding desired outputs, namely, $\{y_1, y_2, ..., y_K\}$. Therefore, the value of $F(\mathbf{P}_i^t)$ directly reflects the effectiveness of ALMMo* on approximating the given (nonlinear) prediction problem.

Then, the individual best position of each particle, $\mathbf{Pb}_i$ is set as: $\mathbf{Pb}_i \leftarrow \mathbf{P}_i^t$ and the global best position, $\mathbf{Gb}$ is selected from the swarm according to the following rule:

$$\mathbf{Gb} \leftarrow \mathbf{P}_{i*}^t; \quad i* = \underset{i=1,2,...,L}{\arg\min}\,(F(\mathbf{P}_i^t)). \tag{27}$$

Stage 2. Particle Updating

In this stage, the algorithm iteratively searches the space for a better solution. The velocity, $\mathbf{v}_i^{t+1}$ of each particle (assuming the i^{th} one, $i = 1, 2, ..., L$) $\mathbf{P}_i^t$ is firstly calculated by the following:

$$\mathbf{v}_i^{t+1} = \omega\mathbf{v}_i^t + c_1 \cdot \mathbf{r}_1 \circ (\mathbf{Pb}_i - \mathbf{P}_i^t) + c_2 \cdot \mathbf{r}_2 \circ (\mathbf{Gb} - \mathbf{P}_i^t); \tag{28}$$

where $\mathbf{v}_i^0 = \mathbf{0}_{N\times 2(M+1)}$; ω is the inertia weight determining how much the current velocity is preserved; c_1 and c_2 are the two acceleration coefficients that determine the relative learning weights of $\mathbf{Pb}_i$ and $\mathbf{Gb}$; and $\mathbf{r}_1$ and $\mathbf{r}_2$ are two randomly generated matrices with the same dimensionality of $\boldsymbol{P}_i^t$ following uniform distribution with the value range of $[0, 1]$.

To prevent a particle from moving too fast, the following constraint is applied:

$$\begin{cases} \mathbf{v}_i^{t+1}(k,j) = v_j^*, & if\ \mathbf{v}_i^{t+1}(k,j) > v_j^*; \\ \mathbf{v}_i^{t+1}(k,j) = -v_j^*, & if\ \mathbf{v}_i^{t+1}(k,j) < -v_j^*; \end{cases} \tag{29}$$

where $\mathbf{v}_i^{t+1}(k,j)$ denotes the element at the k^{th} row j^{th} column of $\mathbf{v}_i^{t+1}$; $v_j^* = 0.1 \cdot \beta_j$. Then, the particle is updated with the velocity using Eqn. (30):

$$\mathbf{P}_i^{t+1} = \mathbf{P}_i^t + \mathbf{v}_i^{t+1}. \tag{30}$$

After this, the fitness of the updated particle, $\mathbf{P}_i^{t+1}$ is calculated using Eqn. (26), and the following two conditions are examined:

$$\begin{aligned} \textbf{Condition 4a}: \ &\textit{If } (F(\mathbf{P}_i^{t+1}) < F(\mathbf{Pb})) \\ &\textit{Then } (\mathbf{Pb}_i \leftarrow \mathbf{P}_i^{t+1}) \end{aligned} \tag{31a}$$

$$\begin{aligned} \textbf{Condition 4b}: \ &\textit{If } (F(\mathbf{P}_i^{t+1}) < F(\mathbf{Gb})) \\ &\textit{Then } (\mathbf{Gb} \leftarrow \mathbf{P}_i^{t+1}) \end{aligned} \tag{31b}$$

Then, the next particle ($i \leftarrow i+1$) is updated using the same algorithmic procedure (Eqn. (28)-(31)). After all the particles have been updated during the current iteration, the algorithm iterates ($t \leftarrow t+1$) until convergence or a certain predefined maximum iteration number is reached.

Stage 3. Fuzzy Rule Base Updating

After the algorithm converges or the maximum iteration number has been reached, the optimization process is completed and the global best position $\mathbf{Gb}$ is used to derived the optimal parameter setting ($\{\boldsymbol{p}\}$, $\{X\}$, $\{S\}$ and $\{\boldsymbol{a}\}$), from data by forming data clouds $\{\mathbf{C}\}$ in the data space. Other related meta-parameters, namely, $\Lambda_j \leftarrow 1$, $\eta_j \leftarrow 1$, $I_j \leftarrow K$ and $\mathbf{\Theta}_j \leftarrow \Omega_o \mathbf{I}_{(M+1)\times(M+1)}$ are reset accordingly for each data cloud, $\mathbf{C}_j$ ($j = 1, 2, ..., N$). The flowchart of the optimization process of the proposed algorithm is presented in Fig. 3.

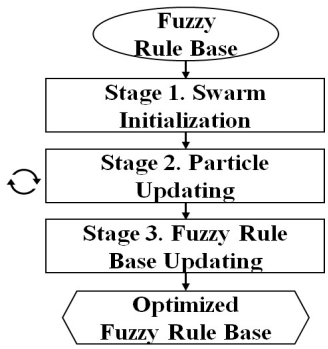

Fig. 3: Flowchart of PSO-based EIS optimization.

Note that the optimization process implemented by the proposed algorithm can be triggered at any point of the online learning process. The ALMMo* system after optimization using past data can continuously learn from new observations in a "one pass" manner as normal. Indeed, it can be optimized for multiple times during the entire learning process if computational resources permit. Nonetheless, without losing generality, the proposed optimization algorithm is only executed at the end of the online learning process unless specifically declared otherwise. In summary, **Algorithm 1** presents the pseudo-code of the proposed procedure for PSO-based ALMMo* optimization.

Algorithm 1 PSO-based ALMMo* optimization.

// ALMMo online learning process //*
while new observation, $\boldsymbol{x}_k$ is available **do**
  if $k = 1$ **then**
    $N \leftarrow 1$
    initialize $\boldsymbol{\mu}$ and χ by (6)
    initialize $\mathbf{C}_N$ by (7)
    initialize $\boldsymbol{a}_N$ and $\mathbf{\Theta}_N$ by (8)
    initialize $\boldsymbol{R}_N$ by (9)
  else
    generate $\hat{y}_k$ by (2)-(4)
    update $\boldsymbol{\mu}$ and χ by (10)
    if Condition 1 is satisfied **then**
      if Condition 2 is satisfied **then**
        update $\mathbf{C}_{n*}$ by (15)
      else
        $N \leftarrow N + 1$
        initialize $\mathbf{C}_N$ by (7)
        initialize $\boldsymbol{a}_N$ and $\mathbf{\Theta}_N$ by (16)
        initialize $\boldsymbol{R}_N$ by (9)
      end if
    else
      update $\mathbf{C}_{n*}$ by (17)
    end if
    remove $\boldsymbol{R}_i$ and $\mathbf{C}_i$ satisfying **Condition 3**
    for $i = 1$ to N **do**
      update $\boldsymbol{a}_i$ and $\mathbf{\Theta}_i$ by (21)
      update $\boldsymbol{R}_i$ with $\boldsymbol{p}_i$ and $\boldsymbol{a}_i$
    end for
  end if
end while
// PSO-based optimization process //
$t \leftarrow 0$
initialize $\mathbf{S}^t$ by (22)-(25)
for $i = 1$ to L **do**
  calculate $F(\mathbf{P}_i^t)$ by (26)
  $\mathbf{Pb}_i \leftarrow \mathbf{P}_i^t$
end for
select $\mathbf{Gb}$ by (27)
while terminal condition is unsatisfied **do**
  for $i = 1$ to L **do**
    update $\mathbf{P}_i^t$ to $\mathbf{P}_i^{t+1}$ by (28)-(30)
    calculate $F(\mathbf{P}_i^{t+1})$ by (26)
    if Condition 4a is satisfied **then**
      $\mathbf{Pb}_i \leftarrow \mathbf{P}_i^{t+1}$
    end if
    if Condition 4b is satisfied **then**
      $\mathbf{Gb} \leftarrow \mathbf{P}_i^{t+1}$
    end if
  end for
end while
derive $\{\mathbf{C}\}$ from $\mathbf{Gb}$
obtain $\{\boldsymbol{p}\}$, $\{X\}$, $\{S\}$ and $\{\boldsymbol{a}\}$
for $i = 1$ to L **do**
  $\Lambda_i \leftarrow 1$
  $\eta_i \leftarrow 1$
  $I_i \leftarrow K$
  $\mathbf{\Theta}_i \leftarrow \Omega_o \mathbf{I}_{(M+1)\times(M+1)}$
  update $\boldsymbol{R}_i$ with $\boldsymbol{p}_i$ and $\boldsymbol{a}_i$
end for

V. Computational Complexity Analysis

A. ALMMo*

The computational complexity of each step of ALMMo* is analyzed here. Since the complexity is dynamically changing subject to the system structure of ALMMo* and the similarity between the current observation and the historical ones, for generality, the analysis is conducted at the time instance when the K^{th} new observation is reached.

Stage 1 concerns system initialization only, and the computational complexity is $O(M)$. In addition, for any $K > 1$, this stage is not performed. The system output is produced at stage 2. The complexity for calculating the firing strengths of the N fuzzy rules is $O(MN)$ thanks to the recursive calculation form (Eqn. (4) and (31)). Thus, the complexity of stage 2 is $O(MN)$ as well. Stage 3 is for updating the global meta-parameters ($\boldsymbol{\mu}$ and χ), and its computational complexity is $O(M)$. Stage 4 involves fuzzy rule base updating. The complexity of computing the global density values at $\boldsymbol{x}_K$ and $\{\boldsymbol{p}\}_N$ is $O(M(N+1))$. The complexity of updating a particular rule or adding a new rule is $O(M)$. Stage 5 is related to fuzzy rule quality monitoring. The most computation costs at this stage take place when calculating the firing strength and thus, the computational complexity of this stage is the same as that of stage 4. Stage 6 requires significantly more computational resources for updating the consequent parameters of fuzzy rules because the FWRLS method needs to update the covariance matrix, see Eqn. (21a). The computational complexity of this stage is $O((M+1)^2N)$. The final stage updates the existing fuzzy rules with the latest premise and consequent parameters, and its computational complexity is negligible.

Therefore, the maximum computational complexity of each processing cycle of ALMMo* is $O((M+1)^2N)$, and the overall computational complexity of the entire learning process is $O((M+1)^2NK)$.

B. Proposed Optimization Algorithm

The computational costs of the proposed optimization algorithm are closely related to the costs of the canonical PSO algorithm. The optimization process involves procedures of initialization, fitness evaluation, and velocity and position updating for each particle. The computational complexity for initializing the swarm is $O((2M+1)NL)$, where L stands for the number of particles. Fitness evaluation for each particle at each iteration cycle has the computational complexity of $O(MNK)$ due to the particular objective function (Eqn. (26)) used. The complexity of velocity and position updating per particle at each iteration cycle is $O((2M+1)N)$ as well. Therefore, assuming that the total iteration number is U, the overall computational complexity of the optimization process using the proposed algorithm is $O(MNKLU)$

VI. Experimental Investigation

In this section, numerical examples are performed for validating the proposed concept and general principles. This experimental study on performance evaluation is based on running the proposed approach over a range of problems, These include: five real-world benchmark regression problems, one Mackey-Glass time series prediction problem, one nonlinear system identification problem and three financial data prediction problems.

The optimized ALMMo* system by the proposed approach is renamed as "particle swarm optimized ALMMo*" (PSO-ALMMo*) for clarity. By default, the parameter setting for the PSO algorithm follows the common practice in the literature [37], namely, $\omega = 0.7298$; $c_1 = c_2 = 1.49618$; the population size is 100 and the maximum iteration number is 200. The reported results in this section are obtained after 25 times Monte Carlo experiments. Details of the comparative approaches involved are summarized in Supplementary Table ST.I. Source code of PSO-ALMMo* is available at: https://github.com/Gu-X.

A. Regression Problems

Five real-world benchmark regression problems, namely, (1) Auto, (2) Autompg, (3) Triazines, (4) Delta Ailerons and (5) California Housing are herein used for performance evaluation. Details of these are summarized in Supplementary Table ST.II. As with the common practice in the research area, all the input and output attributes are normalized in advance to the range of $[0, 1]$.

During each experiment, ALMMo* firstly learns from the training set on a sample-by-sample basis. It is subsequently optimized by the proposed PSO-based algorithm based on all historically collected training data after the online learning process is finished. The performance of PSO-ALMMo* is evaluated on the basis of the testing sets, and the results are reported in Table I in terms of the average prediction accuracy (RMSE with respect to ground truth) and the number of rules derived, denoted by #(Rules) in the table. The curves of RMSE convergence on the five benchmark problems are given in Fig. 4. The prediction results by ALMMo* and ALMMo are also reported in the same table as the baseline under the same experimental protocol. The self-boosting (SB) algorithm [28], which is specifically designed for optimizing EISs, is also involved in this example to better evaluate the effectiveness of the proposed approach. From Table I it can be seen that the proposed optimization algorithm is able to effectively improve the prediction accuracy of ALMMo* on all five benchmark problems. In particular, for Auto and Autompg datasets, the prediction accuracy of the system after the optimization process is improved for more than 20%, and for Triazines dataset, the performance is improved for more than 50%. Moreover,the proposed PSO-based optimization algorithm outperforms the SB algorithm in four out of five cases.

To demonstrate the importance of using ALMMo* to provide the initial parameters for optimization with PSO, dummy PSO-ALMMo* is employed for comparison. Dummy PSO-ALMMo* keeps the same ALMMo* framework, namely, the same number of fuzzy rules, but uses randomly initialized antecedent and consequent parameters for optimization. The results obtained by dummy PSO-ALMMo* (setting 1) are

reported in Table I. In addition, the same experiments are repeated by varying the number of fuzzy rules of dummy PSO-ALMMo* from 3 to 18 (setting 2), and the results are reported in Supplementary Table ST.III. Based on Table I and Supplementary Table ST.III it can be seen that PSO-ALMMo* offers significantly higher performance than dummy PSO-ALMMo* because the parameters learned by ALMMo* provide a good initialization for PSO and expedite it to converge to high-quality solutions.

The prediction performance of PSO-ALMMo* is further compared with the state-of-the-art approaches under the same experimental protocol. The comparison is also reported in Table I. For Trazines, DENFIS [11] fails because of the high-dimensional input attributes and hence, its result is not reported. As can be seen from the comparison, PSO-ALMMo* is able to achieve very high prediction accuracy on the testing data surpassing or, in the worst case, on par with the state-of-the-art approaches.

Apart from the prediction error, statistical significance of the performance improvement of the proposed approach is analyzed in comparison with alternative approaches, by running the statistical pairwise t-tests. In this paper, Fisher's method is employed to combine the p-values returned from the hypothesis tests on the 25 times Monte Carlo experiments:

$$X^2 = -2\sum_{h=1}^{H} \ln(p_h), \tag{32}$$

where p_h is the p-value of the h^{th} hypnosis test ($h = 1, 2, ..., H$); $H = 25$. The X^2 values returned from the t-tests between PSO-ALMMo* and alternatives are presented in Table II, where "Inf" denotes infinite value. Note that the test statistic X^2 tends to be large when the p-values are small, which suggests that the null hypotheses are not true for every test. If the obtained p-values from the 25 hypothesis tests are all greater than 0.05, X^2 is smaller than 149.7866. In addition, pairwise t-tests are also performed between the ground truth and all approaches, and the returned test statistic X^2 values are reported in Table II as well. From Table II it can be seen that the predictions made by PSO-ALMMo* are significantly better than most alternatives and are very close to the ground truth.

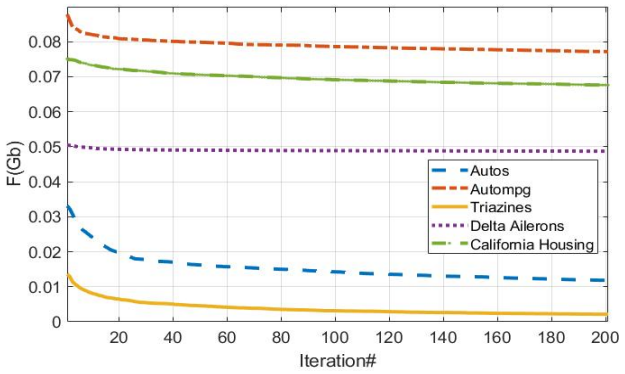

Fig. 4: Convergence curves of proposed optimization algorithm.

To compare between the online learning abilities of ALMMo* and ALMMo, the performances of both algorithms are evaluated by following the typical online learning evaluation method, "test-then-train" on a single epoch using the five benchmark problems as well as three other problems (including a nonstationary one) that are involved in the numerical examples to be presented later, and the results are given in Supplementary Table ST.IV. It is noticeable that the online learning performance of ALMMo* system is stronger than ALMMo thanks to the Gaussian kernel function used.

The same five problems are further considered to assess the learning ability of PSO-ALMMo* by breaking the time dependence between training samples. In this investigation, the order of training samples is randomly scrambled, and the statistical performance of the proposed algorithm is reported in Table III in the form of $mean \pm std$ in terms of prediction RMSE and #(Rules). From Table III it is shown that PSO-ALMMo* outperforms all other approaches in most cases demonstrating that the proposed optimization approach can, indeed, effectively improve the prediction accuracy of ALMMo*.

Numerical experiments are also conducted to evaluate the continuously learning ability of PSO-ALMMo*. The same experimental protocol as above is employed. However, the proposed optimization algorithm is here executed at the middle of the learning process using the first half of the training set. The statistical results are reported in supplementary Table ST.V. Such results illustrate that unlike other approaches, PSO-ALMMo* can continue to learn from new observations in a "one pass" manner and can successfully follow changing data patterns without a full retraining by autonomously gaining new fuzzy rules and discarding stale rules with trivial contributions to the overall system output.

To further demonstrate the merits and generality of the proposed optimization algorithm, it is applied for optimizing ALMMo [7], AnYa [50] and FCMMS [53] following the same experimental protocol as Table III. The more recently introduced GLPSO algorithm [37] is used for optimizing the EISs under the same framework as described in Section IV. The parameter setting of GLPSO follows the mode of [37], namely, $\omega = 0.7298$; $c = c_1 = c_2 = 1.49618$; the population size is 100 and the maximum iteration number is 200. The numerical results are reported in Supplementary Table ST.VI, The results show that the proposed approach can effectively optimize the antecedent and consequent parts of other EISs with similar operating mechanisms, enhancing their prediction performance. Interestingly, such results also reflect the fact that the effectiveness of the proposed optimization approach can be further enhanced by using more advanced PSO variants.

B. Mackey-Glass Time Series Prediction

The proposed PSO-ALMMo* is, in this subsection, tested on the widely-used chaotic Mackey–Glass time series. The detailed problem description is given in Supplementary Material. The prediction accuracy in terms of nondimensional error index (NDEI) and #(Rules) obtained by PSO-ALMMo* and alternative prediction approaches are given in Table IV. It can

TABLE I: PERFORMANCE COMPARISON ON REAL-WORLD BENCHMARK PROBLEMS

Algorithm	Auto		Autompg		Triazines		Delta Ailerons		California Housing	
	RMSE	*#(Rules)*	*RMSE*	*#(Rules)*	*RMSE*	*#(Rules)*	*RMSE*	*#(Rules)*	*RMSE*	*#(Rules)*
PSO-ALMMo*	**0.0425**	8	**0.0694**	7	0.0043	7	**0.0497**	10	**0.0711**	11
ALMMo*	0.0560		0.0858		0.0092		0.0511		0.0791	
Dummy PSO-ALMMo*	0.3902		0.1939		0.7424		0.1004		0.3730	
SB-ALMMo*	0.0457	3	0.0830	2	0.0022	3	0.0506	5	0.0734	5
ALMMo [7]	0.0568	8	0.0842	9	0.0078	7	0.0512	10	0.0777	10
SB-ALMMo [28]	0.0452	4	0.0934	3	**0.0022**	3	0.0512	4	0.0771	5
DENFIS [11]	0.4516	8	0.1458	7	/	/	**0.0497**	11	0.0715	14
CEFNS [12]	0.0666	2	0.0750	2	0.0452	6	0.0502	3	0.0878	2
ESAFIS [51]	0.0604	3	0.0731	33	0.0331	19	0.0506	13	0.0892	6
SAFIS [17]	0.1184	5	0.0993	2	0.0581	9	0.0549	14	0.0988	12
AnYa [50]	0.0604	3	0.0958	2	0.0224	6	0.0515	1	0.0818	3
eTS [14]	0.0535	3	0.0864	6	0.0179	9	0.0513	4	0.0772	3
Simpl_eTS [22]	0.0765	5	0.0765	5	0.0197	9	0.0512	4	0.0773	3
McFIS [52]	0.0689	10	0.0806	4	0.0556	12	0.0509	15	0.0822	15
McIT2FIS [18]	0.0687	3	0.0806	4	0.0556	12	0.0509	15	0.0822	15

TABLE II: p-VALUE IN STATISTICAL PAIRWISE t-TEST ANALYSIS

	Dataset	PSO-ALMMo*	ALMMo*	ALMMo	SBALMMo	SAFIS	ESAFIS	AnYa
PSO-ALMMO* vs	Auto	/	220.1704	226.573	45.457	128.5498	37.9377	263.3305
	Autompg	/	146.2628	122.2057	92.7903	1123.5411	244.7446	1934.1507
	Triazines	/	55.8087	54.4101	52.2966	145.9701	579.4935	858.0194
	Delta Ailerons	/	2157.0497	7519.9293	136.5655	18355.2751	655.2257	Inf
	California Housing	/	236.9164	226.9166	237.8739	Inf	1982.3634	32900.9305
Ground Truth vs	Auto	44.3194	197.1727	202.8881	70.2553	73.5143	49.0860	226.7226
	Autompg	24.6885	36.9609	33.8431	23.4307	553.7204	22.36515	1746.1042
	Triazines	85.4557	81.5286	80.5165	183.0187	139.0642	559.7690	873.5257
	Delta Aileron	60.6014	334.8833	856.8403	88.3148	3889.5226	156.9097	31536.0079
	California Housing	81.8245	69.3474	36.6639	48.2006	10792.2564	669.6971	17622.0537

be seen that PSO-ALMMo* is able to produce the second best prediction result amongst all on this dataset (with the RMSE reduced for more than 55% after optimization). Whilst its system complexity is much lower than that of the approach with the best result.

C. Nonlinear System Identification Problem

This experiment uses a nonlinear system identification problem as a benchmark for further evaluating the performance of PSO-ALMMo*. The problem description is given in Supplementary Material. The performance of all involved approaches are given in Table V in terms of prediction RMSE and #(Rules). It is clearly shown that PSO-ALMMo* surpasses alternatives by producing the most accurate prediction result (with the RMSE dropping for more 60% after optimization). In addition, its system complexity is also at the low end among the comparative algorithms. The identified fuzzy rules during online learning process and the optimized rules after the iterative optimization process by the proposed approach are tabulated in Supplementary Tables ST.VII and ST.VIII, respectively, for better demonstration.

D. Nonstationary Regression Problems

Nonstationary regression problems are typically more challenging than problems of other types. This is because they usually have much more frequent and intensive changes in the data patterns. Such problems are very useful for evaluating the performance of learning algorithms in complex real-world application scenarios. In this paper, three real-world financial data prediction problems are utilized for numerical examples, details of which are given in Supplementary Material.

In the first numerical example, the popular high-frequency trading problem named QuantQuote second resolution market dataset is considered. The data has been standardized online before conducting experiments. The results by PSO-ALMMo* as well as other competitive prediction approaches are reported in Table VI. As can be seen, PSO-ALMMo* produces the best prediction results in all experiments despite that the prediction accuracy of ALMMo* is below the average. This numerical example serves as a strong proof of the effectiveness of the proposed optimization approach.

The second nonstationary problem used for performance evaluation is named S&P 500 closing price prediction. The data has been normalized to the range of [0,1] in advance [60]. The prediction performance in terms of NDEI and #(Rules) obtained by PSO-ALMMo* for the proposed and alternative fuzzy and neuro-fuzzy approaches are tabulated in Table VII, demonstrating that PSO-ALMMo* outperforms all the comparative approaches on this dataset.

The performance of PSO-ALMMo* is further evaluated on the foreign currency exchange rate prediction problem. Note that foreign currency exchange rates usually do not change drastically in a short time. Thus, there is no significant difference between exchange rates of one day and the next day. The statistical performance, in terms of RMSE and #(Rule), of PSO-ALMMo* is reported in Supplementary Table ST.IX, and the results by other approaches are tabulated in the

TABLE III: STATISTICAL RESULTS ON REAL-WORLD BENCHMARK PROBLEMS BY RANDOMLY SCRAMBLING TRAINING SAMPLES

Dataset	Algorithm	RMSE	#(Rules)
Auto	PSO-ALMMo*	**0.0415±0.0038**	8.6±2.1
	ALMMo*	0.0545±0.0023	
	SB-ALMMo [28]	0.0456±0.0002	3.6±1.8
	ALMMo [7]	0.0569±0.0024	7.8±2.5
	FCMMS [53]	0.1067±0.0469	1.2±0.5
	ESAFIS [51]	0.0437±0.0032	2.2±0.4
	SAFIS [17]	0.1752±0.0443	2.4±0.8
	AnYa [50]	0.0624±0.0187	2.6±0.6
	FNN	0.0968±0.0215	/
	LSTM [58]	0.0442±0.0033	/
Autompg	PSO-ALMMo*	**0.0663±0.0025**	8.2±1.3
	ALMMo*	0.0833±0.0024	
	SB-ALMMo [28]	0.0928±0.0003	3.4±1.0
	ALMMo [7]	0.0846±0.0025	8.4±1.8
	FCMMS [53]	0.1504±0.0362	6.0±1.8
	ESAFIS [51]	0.0738±0.0074	2.1±0.2
	SAFIS [17]	0.1352±0.0341	2.5±0.6
	AnYa [50]	0.0988±0.0134	2.0±0.0
	FNN	0.0832±0.0100	/
	LSTM [58]	0.0708±0.0035	/
Triazines	PSO-ALMMo*	0.0041±0.0010	6.9±1.2
	ALMMo*	0.0100±0.0015	
	SB-ALMMo [28]	**0.0021±0.0001**	3.3±1.1
	ALMMo [7]	0.0103±0.0017	6.8±1.8
	FCMMS [53]	0.0343±0.0164	1.0±0.0
	ESAFIS [51]	0.0108±0.0033	2.1±0.3
	SAFIS [17]	0.2212±0.0940	2.6±1.0
	AnYa [50]	0.0745±0.0354	5.9±0.9
	FNN	0.0307±0.0143	/
	LSTM [58]	0.0152±0.0045	/
Delta Ailerons	PSO-ALMMo*	**0.0497±0.0002**	9.7±1.1
	ALMMo*	0.0516±0.0006	
	SB-ALMMo [28]	0.0512±0.0000	4.7±1.2
	ALMMo [7]	0.0515±0.0004	9.3±1.3
	FCMMS [53]	0.0682±0.0197	2.7±2.9
	ESAFIS [51]	0.0509±0.0008	2.2±0.4
	SAFIS [17]	0.0620±0.0092	5.5±1.4
	AnYa [50]	0.0512±0.0000	1.0±0.0
	FNN	0.0514±0.0019	/
	LSTM [58]	0.0511±0.0019	/
California Housing	PSO-ALMMo*	0.0706±0.0010	9.6±1.9
	ALMMo*	0.0779±0.0006	
	SB-ALMMo [28]	0.0771±0.0000	4.1±1.2
	ALMMo [7]	0.0782±0.0006	9.4±1.4
	FCMMS [53]	0.1071±0.0405	3.0±2.4
	ESAFIS [51]	0.0716±0.0022	4.2±0.9
	SAFIS [17]	0.0945±0.0043	10.4±1.7
	AnYa [50]	0.0782±0.0034	2.6±0.6
	FNN	0.0810±0.0161	/
	LSTM [58]	**0.0602±0.0044**	/

same table for comparison. Despite there being no significant difference between the exchange rates of two consecutive days as aforementioned, the performance of ALMMo* is still improved by the proposed optimization algorithm.

E. Remarks

Numerical examples presented in this paper demonstrate that the proposed approach can effectively optimize first-order EISs for better performance. In particular, the following remarks are worth noting:

Firstly, the performance of a first-order EIS depends on many different factors, including model size, rule form (e.g., zero-order, first-order), membership function type (e.g., Gaussian, Cauchy, triangular), and inherent learning mechanisms, etc. An EIS may also behave very differently for different problems depending on their nature.

TABLE IV: PERFORMANCE COMPARISON ON MACKEY-GLASS TIME SERIES PROBLEM

Algorithm	NDEI	#(Rules)
PSO-ALMMo*	0.191	8
ALMMo*	0.432	
SB-ALMMo [28]	0.437	5
ALMMo [7]	0.402	9
DENFIS [11]	0.276	58
CEFNS [12]	0.303	8
ESAFIS [51]	0.312	10
SAFIS [17]	0.380	21
AnYa [50]	0.501	1
eTS [14]	0.356	99
Simpl_eTS [22]	0.376	21
FLEXFIS [16]	**0.157**	89
PANFIS [20]	0.301	19
GENEFIS (C) [21]	0.280	19
GENEFIS (B) [21]	0.339	9
eT2RFNN [54]	0.320	3
GSETSK [55]	0.347	19
SPLAFIS [56]	0.279	30
LEOA [25]	0.248	42

TABLE V: PERFORMANCE COMPARISON ON NONLINEAR SYSTEM IDENTIFICATION PROBLEM

Algorithm	RMSE	#(Rules)
PSO-ALMMo*	**0.0148**	7
ALMMo*	0.0383	
SB-ALMMo [28]	0.1267	3
ALMMo [7]	0.1159	8
DENFIS [11]	0.2599	3
CEFNS [12]	0.0183	12
ESAFIS [51]	0.0338	15
SAFIS [17]	0.0221	17
AnYa [50]	0.0546	2
eTS [14]	0.0212	49
Simpl_eTS [22]	0.0225	22
FLEXFIS [16]	0.0171	8
SOFMLS [57]	0.0201	5

Secondly, PSO, in general, produces better results on the first-order EISs with a smaller model size because of the lower dimensionality of the problem space (see Supplementary Table ST.VI). This is also observed from numerical examples presented in [26], where PSO is employed for zero-order EIS optimization.

VII. CONCLUSION

This paper presents a PSO-based approach for first-order EIS optimization. The proposed approach is generic and purely data-driven. It simultaneously optimizes the antecedent and consequent parameters of EISs, learning the optimal models based on historically observed data. Moreover, the proposed approach does not change the underlying operating mechanisms and learning behaviors of EISs. The optimized system model can continuously self-update from new observations in a "one pass" manner without a full retraining. Experimental

TABLE VI: PERFORMANCE COMPARISON ON QUANTQUOTE DATASET

Dataset	Algorithm	NDEI	#(Rules)
$x_{k+10,4} = f(\boldsymbol{x}_k)$	PSO-ALMMo*	**0.3351±0.1756**	8.6±2.4
	ALMMo*	0.5983±0.6200	
	SB-ALMMo [28]	0.3361±0.1864	4.8±1.6
	ALMMo [7]	0.6131±0.5947	9.1±2.4
	FCMMS [53]	0.4147±0.3129	6.2±3.0
	ESAFIS [51]	0.4338±0.2659	2.3±0.9
	SAFIS [17]	0.8337±0.6703	22.2±7.4
	AnYa [50]	0.9369±0.7445	4.5±0.8
	FNN	0.5784±0.8356	/
	LSTM [58]	0.3456±0.1662	/
$x_{k+15,4} = f(\boldsymbol{x}_k)$	PSO-ALMMo*	**0.4232±0.1534**	9.4±2.7
	ALMMo*	1.0044±0.9079	
	SB-ALMMo [28]	0.4350±0.1687	4.6±2.3
	ALMMo [7]	0.8558±0.6770	8.9±2.2
	FCMMS [53]	0.5592±0.3541	6.4±3.6
	ESAFIS [51]	1.0487±1.0100	2.2±1.1
	SAFIS [17]	1.7274±1.3923	19.9±8.9
	AnYa [50]	1.0192±1.1586	4.8±0.7
	FNN	1.2029±2.4667	/
	LSTM [58]	0.4336±0.1494	/
$x_{k+20,4} = f(\boldsymbol{x}_k)$	PSO-ALMMo*	**0.5278±0.1963**	8.7±1.9
	ALMMo*	1.3637±1.6079	
	SB-ALMMo [28]	0.5482±0.2338	4.5±2.3
	ALMMo [7]	1.2946±1.3748	8.4±2.0
	FCMMS [53]	0.6169±0.3128	6.0±2.9
	ESAFIS [51]	1.3302±1.4781	2.8±1.1
	SAFIS [17]	1.9117±1.7789	22.5±18.8
	AnYa [50]	1.0751±0.9620	4.3±0.7
	FNN	0.7968±1.1031	/
	LSTM [58]	0.5329±0.2005	/

TABLE VII: PERFORMANCE COMPARISON ON S&P 500 DATASET

Algorithm	NDEI	#(Rules)
PSO-ALMMo*	**0.0146**	6
ALMMo*	0.0147	6
ALMMo [7]	0.0149	7
PANFIS [20]	0.09	4
GENEFIS [21]	0.07	2
eT2RFNN [54]	0.04	2
ANFIS [10]	0.02	32
LEOA [25]	0.1229	52
SEFS [59]	0.0182	2
EFS-SLAT [60]	0.0156	23

investigation with a range of real-world data-based benchmark problems demonstrate that the proposed optimization approach effectively improves the overall performance of EISs.

There are several considerations for further works Firstly, it is interesting to examine and introduce any new mechanism to monitor the quality of identified fuzzy rules during the optimization process. By removing useless fuzzy rules, efficiency can be gained over the computation- and memory- resources required. Secondly, the optimization process is conveniently conducted at the end of the online learning process, it would be very useful to develop certain criteria that can help EISs to self-determine when to carry out system optimization to ensure the high system performance being maintained. Another important direction is to consider alternative EC techniques (e.g., genetic algorithms) for EIS optimization. Different EC techniques have different pros and cons, it may be beneficial to investigate how they perform comparing with the use of PSO.

REFERENCES

[1] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, " Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Information Sciences*, vol. 490, pp. 344-368, 2019.

[2] E. Lughofer, "Evolving fuzzy systems–fundamentals, reliability, interpretability, useability, applications," in *Handbook on Computational Intelligence*, P. Angelov, Ed. New York: World Scientific, 2016, pp. 67–135.

[3] L. Maciel, R. Ballini, and F. Gomide, "Evolving possibilistic fuzzy modeling for realized volatility forecasting with jumps," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 2, pp. 302–314, 2017.

[4] A. B. Asghar and X. Liu, "Adaptive neuro-fuzzy algorithm to estimate effective wind speed and optimal rotor speed for variable-speed wind turbine," *Neurocomputing*, vol. 272, pp. 495–504, 2018.

[5] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[6] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *IEEE international conference on neural networks*, 1995, pp. 1942–1948.

[7] P. P. Angelov, X. Gu, and J. C. Principe, "Autonomous learning multimodel systems from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 2213–2224, 2018.

[8] I. Škrjanc, G. Andonovski, A. Ledezma, O. Sipele, J. A. Iglesias, and A. Sanchis, "Evolving cloud-based system for the recognition of drivers' actions," *Expert Systems with Applications*, vol. 99, pp. 231–238, 2018.

[9] N. Anh, S. Suresh, M. Pratama, and N. Srikanth, "Interval prediction of wave energy characteristics using meta-cognitive interval type-2 fuzzy inference system," *Knowledge-Based Systems*, vol. 169, pp. 28–38, 2019.

[10] J.S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol.23 no.3, pp. 665-685, 1993.

[11] N. K. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[12] R. Bao, H. Rong, P. P. Angelov, B. Chen, and P. K. Wong, "Correntropy-based evolving fuzzy neural system," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1324–1338, 2018.

[13] M. Pratama, W. Pedrycz, and E. Lughofer, "Evolving ensemble fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2552–2567, 2018.

[14] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*: *Cybernetics*, vol. 34, no. 1, pp. 484–498, 2004.

[15] M. Pratama, S. G. Anavatti, M. J. Er, and E. D. Lughofer, "pClass: an effective classifier for streaming examples," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 369–386, 2015.

[16] E. D. Lughofer, "FLEXFIS: a robust incremental learning approach for evolving Takagi-Sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[17] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[18] K. Subramanian, A. K. Das, S. Sundaram, and S. Ramasamy, "A meta-cognitive interval type-2 fuzzy inference system and its projection based learning algorithm," *Evolving Systems*, vol. 5, no. 4, pp. 219–230, 2014.

[19] M. Pratama, J. Lu, and G. Zhang, "Evolving type-2 fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 574–589, 2016.

[20] M. Pratama, S.G. Anavatti, P.P. Angelov, and E. Lughofer, "Panfis: a novel incremental learning machine," *IEEE Transactions on Neural Network and Learning Systems*, vol. 25, no. 1, pp. 55-68, 2014.

[21] M. Pratama, S. G. Anavatti, and E. Lughofer,"Genefis: toward an effective localist network," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 547–562, 2014.

[22] P. Angelov and D. Filev, "Simpl˙eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *IEEE International Conference on Fuzzy Systems*, 2005, pp. 1068–1073.

[23] R. M. Johnstone, C. Richard Johnson, R. R. Bitmead, and B. D. O. Anderson, "Exponential convergence of recursive least squares with exponential forgetting factor," *Systems and Control Letters*, vol. 2, no. 2, pp. 77–82, 1982.

[24] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2057–2068, 2011.

[25] D. Ge and X. J. Zeng,"Learning evolving T-S fuzzy systems with both local and global accuracy - a local online optimization approach," *Applied Soft Computing*, vol. 86, pp. 795–810, 2018.

[26] X. Gu, P. Angelov and H. J. Rong, "Local optimality of self-organising neuro-fuzzy inference systems," *Information Sciences*, vol. 503, pp. 351-380, 2019.

[27] H. J. Rong, P. Angelov, X. Gu, and J. M. Bai, "Stability of evolving fuzzy systems based on data clouds," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2774–2784, 2018.

[28] X. Gu and P. Angelov, "Self-boosting first-order autonomous learning neuro-fuzzy systems," *Applied Soft Computing*, vol. 77, pp. 118–134, 2019.

[29] X. Xia, C. Xie, B. Wei, Z. Hu, B. Wang, and C. Jin, "Particle swarm optimization using multi-level adaptation and purposeful detection operators," *Information Sciences*, vol. 385–386, pp. 174–195, 2017.

[30] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[31] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi, "Fuzzy self-tuning PSO: a settings-free algorithm for global optimization," *Swarm and Evolutionary Computation* vol. 39, pp. 70–85, 2018.

[32] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*: *Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[33] K. Chen, F. Zhou, L. Yin, S. Wang, Y. Wang, and F. Wan, "A hybrid particle swarm optimizer with sine cosine acceleration coefficients," *Information Sciences*, vol. 422, pp. 218–241, 2018.

[34] H. L. Shieh, C. C. Kuo, and C. M. Chiang,"Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification," *Applied Mathematics and Computation*, vol. 218, no. 8, pp. 4365–4383, 2011.

[35] H. Wang, Y. Jin, and J. Doherty, "Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, 2017.

[36] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *IEEE Swarm Intelligence Symposium*, 2003, pp. 174–181.

[37] Y. J. Gong, J. J. Li, Y. Zhou, Y. Li, et al., "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*., vol. 46, no. 10, pp. 2277–2290, 2016.

[38] A. Lin, W. Sun, H. Yu, G. Wu, and H. Tang,"Global genetic learning particle swarm optimization with diversity enhancement by ring topology," *Swarm and Evolutionary Computation*, vol. 44, 2018, pp. 571–583, 2019.

[39] M. Alswaitti, M. Albughdadi, and N. A. M. Isa, "Density-based particle swarm optimization algorithm for data clustering," *Expert Systems with Applications*, vol. 91, pp. 170–186, 2018.

[40] A. Moharam, M. A. El-Hosseini, and H. A. Ali, "Design of optimal PID controller using hybrid differential evolution and particle swarm optimization with an aging leader and challengers," *Applied Soft Computing*, vol. 38, pp. 727–737, 2016.

[41] H. Yang, Q. Du, and G. Chen,"Particle swarm optimization-based hyperspectral dimensionality reduction for urban land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 544-554, 2012.

[42] P. Ghamisi and J. A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 309–313, 2015.

[43] C .F. Juang, C. M. Hsiao, and C. H. Hsu, "Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 1, pp. 14–26, 2010.

[44] T. Chen, Q. Shen, S. Pan and C. Shang, "Fuzzy rule weight modification with particle swarm optimisation," *Soft Computing*, vol. 20, no. 8, pp. 2923-2937, 2016.

[45] E. Camci, D. R. Kripalani, L. Ma, E. Kayacan and M. A. Khanesar, "An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 1-8, 2018.

[46] S. Mirjalili, S. Z. Mohd Hashim, and H. Moradian Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11125-11137, 2012.

[47] A. Okabe, B. Boots, K. Sugihara and S. Chiu, "Spatial tessellations: concepts and applications of Voronoi diagrams," *John Wiley and Sons*, 2009.

[48] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[49] S. Saremi, S. Mirjalili, A. Lewis, A. W. C. Liew, and J. S. Dong, "Enhanced multi-objective particle swarm optimisation for estimating hand postures," *Knowledge-Based Systems*, vol. 158, pp. 175–195, 2018.

[50] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *International Journal of General Systems*, vol. 41, no. 2, pp. 163–185, 2012.

[51] H. J. Rong, N. Sundararajan, G. B. Huang and G.S. Zhao "Extended sequential adaptive fuzzy inference system for classification problems," *Evolving Systems*, vol.2, no. 2, pp. 71-82, 2011.

[52] K. Subramanian and S. Suresh, "A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system," *Applied Soft Computing*, vol. 12, no.11, pp. 3603–3614, 2012.

[53] P. Angelov, "Fuzzily connected multimodel systems evolving autonomously from data streams," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 4, pp. 898-910, 2011.

[54] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M.J. Er, " An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks," *IEEE Transactions on fuzzy system*, vol.25, no.5, pp. 1175-1192, 2017.

[55] N.N. Nguyen, W.J. Zhou, and C. Quek,"Gsetsk: a generic self-evolving tsk fuzzy neural network with a novel Hebbian-based rule reduction approach," *Applied Soft Computing*, vol.35, pp. 29-42, 2015.

[56] R.J. Oentaryo, M.J. Er, S. Linn, and X. Li, "Online probabilistic learning for fuzzy inference system", *Expert Systems with Applications*, vol. 41, no.11, pp. 5082-5096, 2014.

[57] J. de Jesus Rubio,"SOFMLS: online self-organizing fuzzy modified leastsquares network," *IEEE Transactions on fuzzy systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[58] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of machine learning research*, vol. 3, no. 1, pp. 115-143, 2002.

[59] D. Ge and X. J. Zeng,"A self-evolving fuzzy system which learns dynamic threshold parameter by itself," *IEEE Transactions on fuzzy systems*, vol. 27, no. 8, pp. 1625-1637, 2019.

[60] D. Ge and X. J. Zeng,"Learning data streams online- an evolving fuzzy system approach with self-learning/adaptive thresholds," *Information Sciences*, vol. 507, pp. 172-184, 2020.